

J2ME Bluetooth programming

■ ■ ■

Muhammad Allam

Electronic devices connect to one another in a variety of ways where the most famous wireless technologies are infrared and Bluetooth. Infrared is the technology that you can see in TV remote controls or air conditioner remotes where the communication should be pointed to the target device. WiFi technology is used for strong and wide area communication where wireless communication can be made.

A cable connects a computer's processing unit to a display, a data cable and a docking cradle connect a personal digital assistant (PDA) or a cellular phone to a computer, radio waves connect a cordless phone to its base unit, and an infrared beam connects a remote control to a television. That's where Bluetooth comes in.

Bluetooth technology is the most used temporary communication technology, Especially inside mobile devices, palm tops, pocket PCs, and so forth. It can be used to exchange objects, packets, or a simple stream

Contents at a Glance

- What is Bluetooth.....4
- Bluetooth Network Topology.....5
- Bluetooth Communication Types.....6
- What's JABWT & JSR82.....6
- Bluetooth stack layer.....7
- Code and Explanation.....8
- The last word.....13

What's Bluetooth?

Bluetooth is a wireless communication protocol that Ericsson Mobile Communications started developing the Bluetooth system in 1994, looking for a replacement to the cables connecting cellphones and their accessories. It supports distances equal up to 30 feet, or to 10 meters but consumes less power, and transmits at lower speeds. Bluetooth is very attractive to phone and PDA manufacturers due to its low power consumption rate.

The Bluetooth system is named after a tenth-century Danish Viking king, Harald Blåtand, who united and controlled Norway and Denmark. The first Bluetooth devices hit the market around 1999.

Bluetooth vs. Infrared

Home electronics devices like TVs and VCRs communicate using beams of light in the infrared spectrum. Infrared is fairly reliable and doesn't cost much to build into devices. It does have drawbacks

- 1- Its line of sight, so sender must align with its receiver.
- 2- Its one to one, so a device can't send to multiple receivers at the same time.

Bluetooth vs. 802.11b

IEEE 802.11b connects devices in a wireless local area network (LAN), while Bluetooth connects devices in a Personal Area Network (PAN).

To know about it visit <http://wifinetnews.com/archives/001166.html>

Bluetooth Features

The major features of Bluetooth are:

- 1-Bluetooth is wireless and automatic. You don't have to keep track of cables, connectors, and connections, and you don't need to do anything special to initiate communications. Devices find each other automatically and start conversing without user input, except where authentication is required; for example, users must log in to use their email accounts.
- 2-Bluetooth is inexpensive. Market analysts peg the cost to incorporate Bluetooth technology into a PDA, cell phone, or other product at around \$20 now, and say that it could fall to as little as \$5 per unit.

3-Bluetooth handles both data and voice. Its ability to handle both kinds of transmissions simultaneously makes possible such innovations as a mobile hands-free headset for voice with applications that print to fax, and that synchronize the address books on your PDA, your laptop, and your cell phone.

Signals are omni-directional and can pass through walls and briefcases. Communicating devices don't need to be aligned and don't need an unobstructed line of sight.

Bluetooth Network Topology

When two or more Bluetooth-enabled devices come within range and establish a connection, a personal area network is formed. A personal area network can either be a piconet or a scatternet.

Type	Meaning in life
Piconet	Is a single master and up to seven slaves. No matter what kind Bluetooth devices are involved (whether they are phones, access points, PDAs, laptops, headsets, etc.) The master of the piconet is the one that initiates the connection. The device that accepts the connection automatically becomes the slave. Master/slave roles are not predefined; so if a piconet exists between a laptop and a PDA, either device could be the master or the slave.
Scatternet	Is a scatternet is essentially a chain of piconets. A scatternet is formed when the slave of a piconet also become the master of its own piconet.

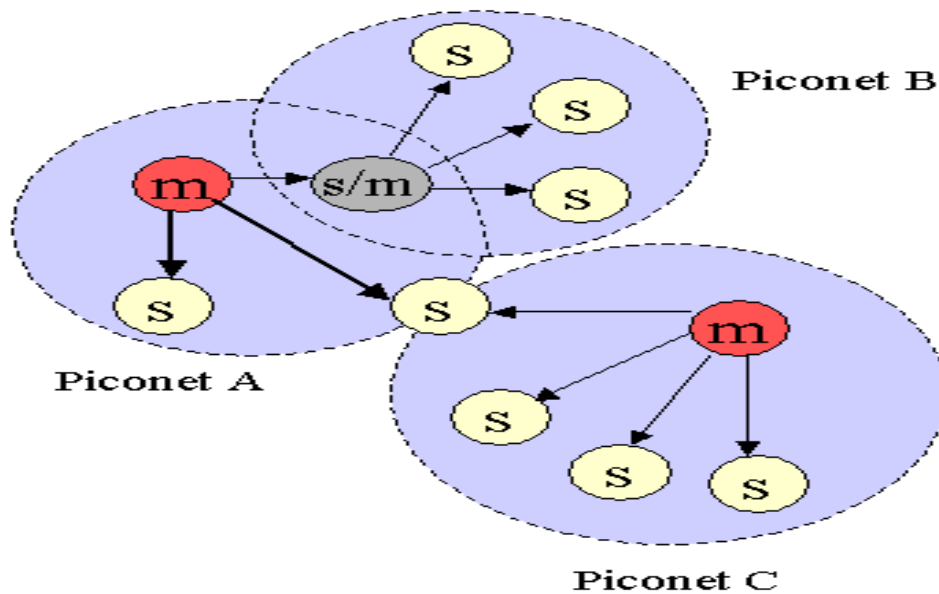


Figure 1.0 show the Piconet and Scatternet.

Bluetooth Communication Types

There are three types of communication protocols defined inside Bluetooth technology:

1-**OBEX**: The "**Ob**ject **Ex**change" communication Protocol is used to Exchange Physical data such as Files, images, and so on in binary Format.

2-**L2CAP**: The "**L**ogical **L**ink **C**ontrol and **A**daptation **P**rotocol" used to send Packets between host and client.

3-**RFCOM**: The "**R**adio **F**requency **C**OMMunication" is very easy and not Complicated .it is used to stream simple data.

Java Bluetooth API (**JABWT**)

Sun Java has introduced the [Bluetooth JSR82 API package](http://jcp.org/aboutJava/communityprocess/final/jsr082/).

<http://jcp.org/aboutJava/communityprocess/final/jsr082/> The JSR82 API has capability to provide all three kinds of communications: Obex, L2CAP, or RFCOMM.

Package	Provides
javax.bluetooth	The core Bluetooth API.
javax.obex	The Object Exchange (OBEX) API.

Table 1.1 JABWT packages

*So what is the JSR-82?

The JSR-82 is the official Java Bluetooth API, which was standardized by Motorola and the following companies in the Java Community Process:

- Sun Microsystems
- IBM
- Nokia.
- Sony Ericsson.
- Symbian.

You can find a list of commercially available [JSR-82 device](http://www.javabluetooth.com/jsr82devices.html) <http://www.javabluetooth.com/jsr82devices.html> on our JSR-82 Devices page.

The Bluetooth protocol stack

What's the purpose of the Bluetooth stack? The stack is the piece of software (and firmware) that controls your Bluetooth device. Figure 2 details the protocol stack.

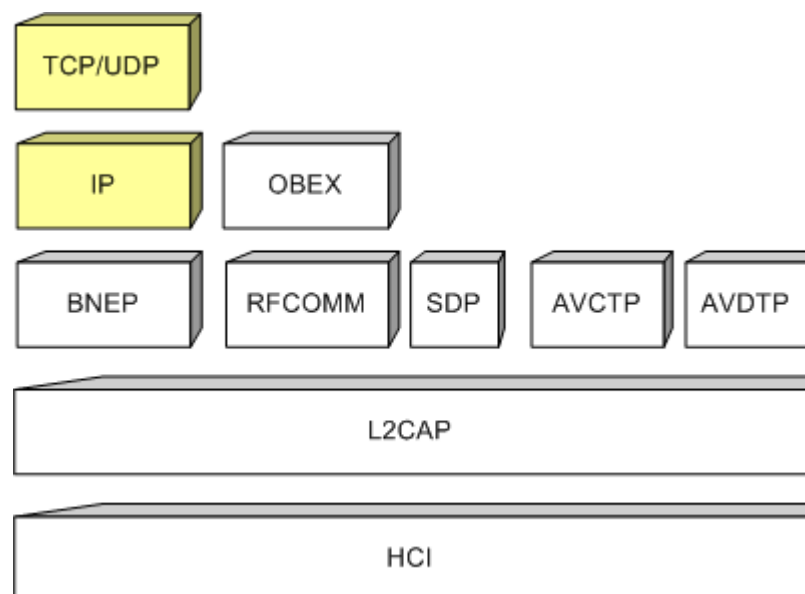


Figure 2. The Bluetooth protocol stack

- The bottom layer of the stack is the **HCI**, the *Host Controller Interface*. This layer is literally the interface between the host (your computer or Hardware) and the controller (the Bluetooth device). As you can see, all other layers go through the HCI.
- The layer above the HCI is **L2CAP**, the *Logical Link Controller Adaptation Protocol*. This layer acts as the data multiplexer for all other layers.

- The next layer is **BNEP**, the *Bluetooth Network Encapsulation Protocol*. Using BNEP, you can run other networking protocols, such as IP, TCP, and UDP, over Bluetooth.
- **RFCOMM** is known as the *virtual serial port protocol* because it allows a Bluetooth device to simulate the functions of a serial port.
- The **OBEX** protocol layer is implemented above the RFCOMM layer and is useful when you want to transfer data as an object, such as files.
- **SDP** is the *Service Discovery Protocol layer*, which is used whenever you want to find services on a remote Bluetooth device.
- The final two layers, **AVCTP** and **AVDTP** are used for the control and distribution of audio and video over Bluetooth. AVCTP and AVDTP are relatively new additions to the Bluetooth protocol; they are used when you want to control the functions of a media player or if you want stream audio in stereo.

Code and Explanation

1-Device discovery(Client)

Is the first step required when browsing nearby Bluetooth devices . When we have discovered nearby devices we can find out which services they offer.

Package	Meaning in life
<code>javax.bluetooth.LocalDevice</code>	Represents My own Bluetooth device.
<code>javax.bluetooth.LocalDevice</code>	Represents all remote Bluetooth devices are represented.
<code>javax.bluetooth.DiscoveryAgent</code>	Is a helpful class that allows you to discover remote Bluetooth devices in the vicinity and returns a <code>javax.bluetooth.RemoteDevice</code> for each Bluetooth device in the area. Search for services in the remote devices that you've found.

Table 1.2 represents common packages.

Let's take a look about device and service discovery code.

```

/* The DiscoveryListener interface allows an application to receive *device
discovery and service discovery events. This interface *provides four methods,
two for discovering devices and two for *discovering services.*/

```

```

public class CoreOperation implements DiscoveryListener{

    /*Represents My own Bluetooth device */
    private LocalDevice local = null;

    /*To discover remote Bluetooth devices in the vicinity and Search
    * for services in The remote devices that you've found. */
    private DiscoveryAgent agent = null;

    Vector devicesFound = null;

    /*To store services records that found at device discovered*/
    private ServiceRecord[] servicesFound = null;

    /*DeviceClass is represent Class of device record that defined by
    * Bluetooth specification .this record contain info about type of
    * the device and type of services available on the device.*/

    public void deviceDiscovered(RemoteDevice remoteDevice,
        DeviceClass deviceClass) {

        devicesFound.addElement(remoteDevice);

    }

    /*@param discType represent the type of request that was completed;
    * either INQUIRY_TERMINATED, INQUIRY_ERROR or INQUIRY_COMPLETED */

    public void inquiryCompleted(int discType) {

        switch (discType) {
            case DiscoveryListener.INQUIRY_COMPLETED:
                /*Search for services.*/
                searchServices ();

                break;

            case DiscoveryListener.INQUIRY_ERROR:

                break;

            case DiscoveryListener.INQUIRY_TERMINATED:

                break;
        }
    }

```

```

}

/*called when service(s) are found during a service search.
 * TransID - the transaction ID of the service search that is
 * posting the result service - a list of services found during
 * the search request
 * ServiceRecord is interface that describe the characteristics of a
 * Bluetooth service .service record contains set of attributes
 * where each attr have ID & value*/
public void servicesDiscovered(int transID, ServiceRecord[] serviceRecord) {
    servicesFound = serviceRecord;
}

/* Called when a service search is completed or was terminated
 * because of an error. Legal status values in the respCode argument include:
 * SERVICE_SEARCH_COMPLETED, SERVICE_SEARCH_TERMINATED,
 * SERVICE_SEARCH_ERROR, SERVICE_SEARCH_NO_RECORDS and
 * SERVICE_SEARCH_DEVICE_NOT_REACHABLE */

public void serviceSearchCompleted(int transID, int respCode) {
}

private void doDeviceDiscovery() {
    try {

        local = LocalDevice.getLocalDevice();

    }catch (BluetoothStateException bse) {
// Error handling code here
    }
    agent = local.getDiscoveryAgent();

    devicesFound = new Vector();

    try {
        /* Places the device into inquiry mode.
        *@param 1st argument is the type of inquiry to
        * complete mean that The inquiry access code for
        * General/Unlimited Inquiry Access Code */

        if(!agent.startInquiry(DiscoveryAgent.GIAC,this)) {
// Inquiry not started, error handling code here
        }
    }catch(BluetoothStateException bse) {
// Error handling code here
    }
}
}

```

```

private void searchServices() throws BluetoothStateException {
    //System.out.println("Size of devices vector is "+devices.size());
    for (int i = 0; i < devices.size(); i++) {
        RemoteDevice rd = (RemoteDevice) devices.elementAt(i);

        try {

            discoveryAgent.searchServices(attrSet, uuidSet,
                rd, this);
        } catch (BluetoothStateException e) {
            System.err.println("Can't search services for:"
                + rd.getBluetoothAddress() + " due to " + e);

            throw new BluetoothStateException(e.getMessage());
        }
    }
}

```

What this line mean `discoveryAgent.searchServices(attrSet, uuidSet, rd, this)?`
`discoveryAgent` object enable me to perform device and service discovery.

searchServices → Searches for services on a remote Bluetooth device that have all the UUIDs specified in `uuidSet`. Once the service is found, the attributes specified in `attrSet` and the default attributes are retrieved.

Parameters:

`attrSet` - indicates the attributes whose values will be retrieved on services which have the UUIDs specified in `uuidSet`

`uuidSet` - the set of UUIDs that are being searched for; all services returned will contain all the UUIDs specified here

`btDev` - the remote Bluetooth device to search for services on

`discListener` - the object that will receive events when services are discovered

Returns:

the transaction ID of the service search; this number must be positive

2-Open connection(Server)

After device discovery and service discovery are completed a client has all the information needed to set up a communication link to a service on the server. To open the Bluetooth connection, you have to build a Bluetooth URL string that will be called inside the `Connector.open(URL)` function; this function will return the `StreamConnectionNotifier` Object. The URL actually is the way to initialize the communication protocol for Bluetooth, just like on an Internet Explorer search box. You just type `http://www.address.com`, where `http://` is the protocol and the rest is the address of the target place. In Bluetooth, you will do something like this:

In REFCOMM:

URL = "btspp://localhost:" + UUID + ";name=rftcommtest;authorize=true"

In L2CAP

URL = " btl2cap://localhost:" + UUID + ";name=rftcommtest;authorize=true"

```
notifier = (StreamConnectionNotifier) Connector.open(  
    url.toString() );
```

There are three versions of the open() method of the Connector class. All three versions require at least a connection string parameter that identifies the connection and throws an IOException error. They also return a Connection. Here's the syntax for the first version of the open() method:

static Connection open (String connectString) throws IOException

The second version of the open() method :

static Connection open (String connectString, int mode) throws IOException

Requires a second parameter, which is an int that identifies the mode in which the connection is opened.

Mode	Description
READ	Open connection for read only
WRITE	Open connection for write only
READ_WRITE	Open connection for read and write

The third version of the open() method:

static Connection open (String connectString, int mode, boolean timeouts) throws IOException

Requires a third parameter consisting of a boolean value that indicates whether or not the application can handle a timeout exception. A timeout exception is thrown whenever an attempt to create the connection fails. The timeout period and other aspects of the timeout exception are implementation dependent.

Finally using notifier object to call acceptAndOpen method that will start waiting for the client connection.

The last word

To get clear view see BluetoothDemo that's a simple chat program over RFCOMM Bluetooth protocol

Happy coding ;-)

Resources

*[Simple Bluetooth Communication in J2ME](#)

<http://www.codeguru.com/java/article.php/c13147/>

*[The Java APIs for Bluetooth Wireless Technology](#)

<http://developers.sun.com/techttopics/mobility/midp/articles/bluetooth1/>

* [File transfer with JSR-82 and OBEX](#)

<http://www-128.ibm.com/developerworks/wireless/library/wi-boogie1/>

*[Java Bluetooth](#)

<http://www.javablueetooth.com/>

*J2ME the Complete Reference by James Keogh.

*J2ME Bluetooth Programming by Master's Thesis.

*J2ME Bluetooth documentation.